Account Management

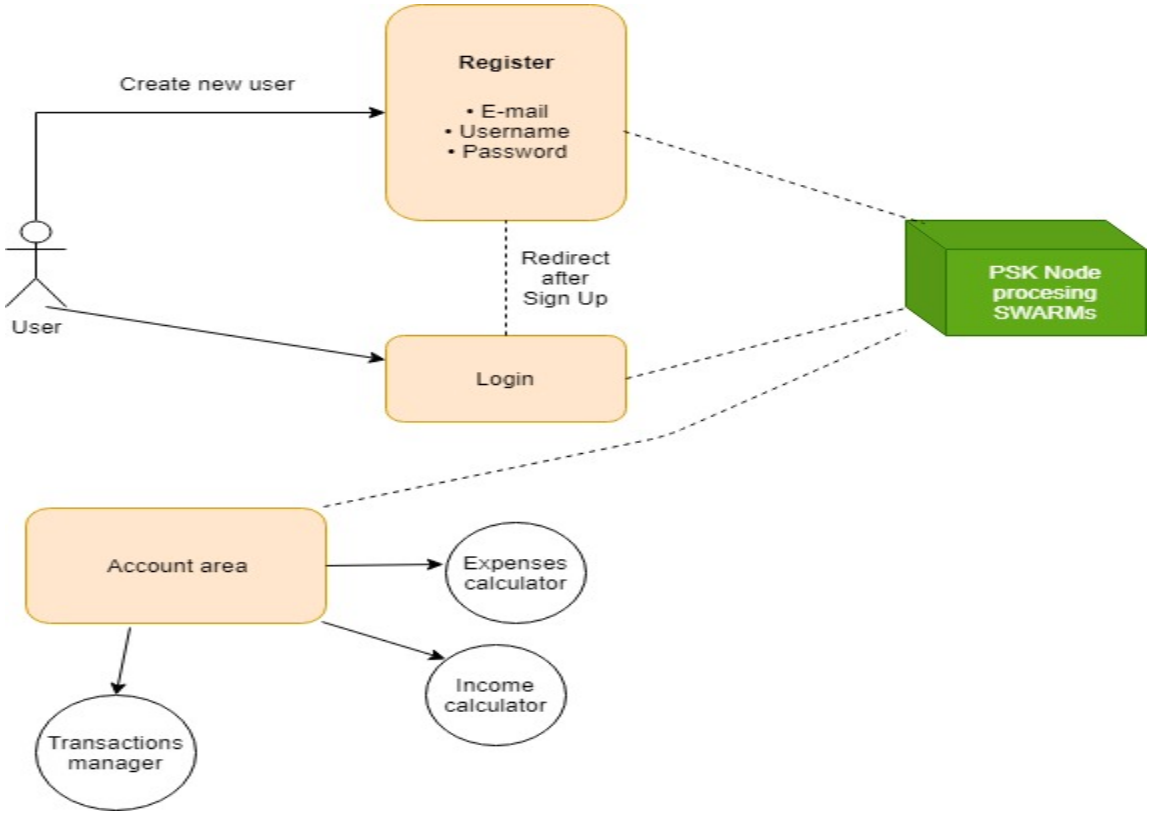
- Client application for Private Sky backend –

This application uses *Private Sky's* **SWARM**s to provide a solution for the client that offers the possibility to create a personal account, log in to it and manage and track the balance of the owned account, through incomes and expenses.

These functionalities are achieved through the created swarms that are integrated into *Private Sky* as extension libraries. This way our client application uses the PSK back-end.

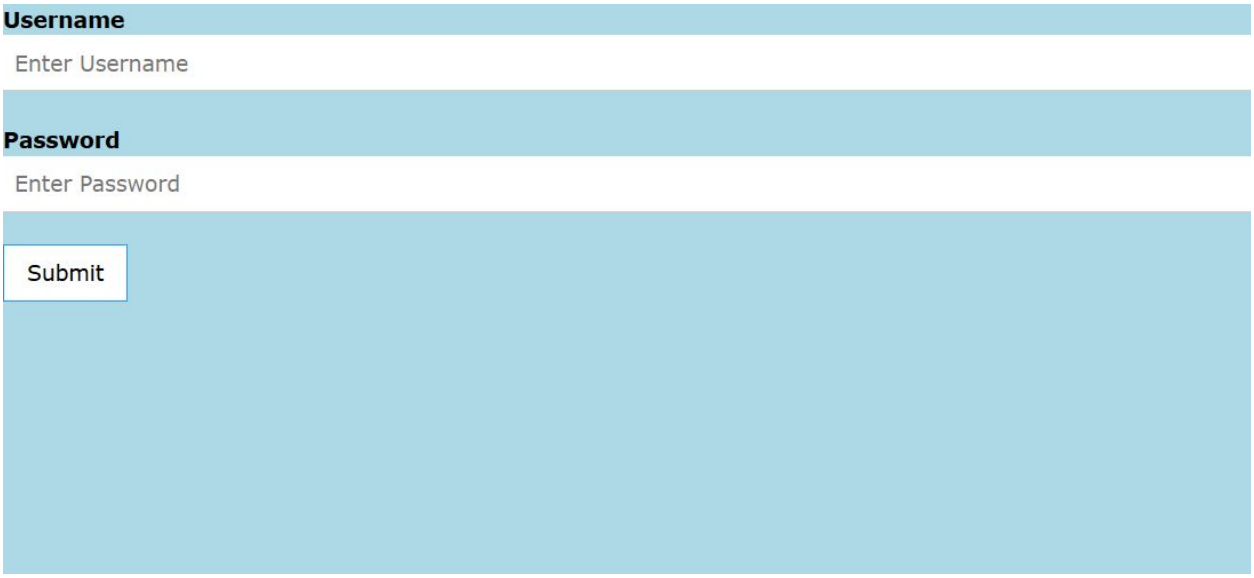
This client app also offers the possibility to transfer different currency between multiple accounts.

Below is an architectural diagram representing the main components of the client application.



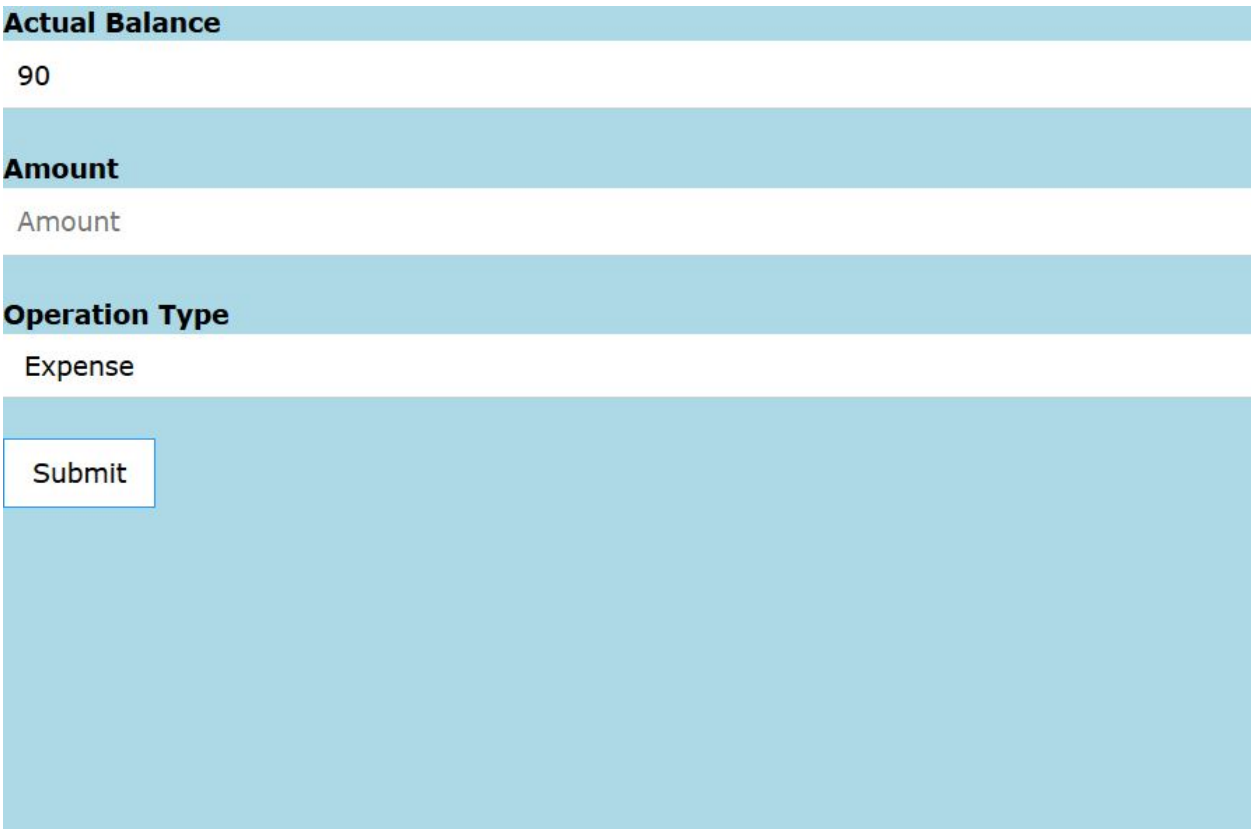
Register page:

Index / Login page:



Account page – before expense operation:

Account page – after expense operation:



Appendix (code snippets)

Code processing the account balance (incomes / expenses) – *addOperation.js*:

```
1.  function addOperation(){
2.      amount = document.getElementById("amountV");
3.      operationType = document.getElementById("operationType");
4.      uname =  document.getElementById("sessionId");
5.
6.      var operationData = {
7.          uname: sessionId.value,
8.          amount: amount.value,
9.          operationType: operationType.value
10.     }
11.
12.     const interact = pskclientRequire("interact");
13.     interact.enableRemoteInteractions();
14.     const ris = interact.createRemoteInteractionSpace('testRemote', 'http://127.0.0.1:8080',
    'local/agent/expense');
15.
16.     ris.startSwarm('Register1', 'addOperation', operationData).onReturn(function(response)
     {
17.         console.log(`Returning message "${response.message}"` );
18.         if(response.succes === true){
19.             var uname = response.uname;
20.             var q = "?user=" + uname + "&balance=" + response.balance;
21.             window.location.href = "account.html" + q;
22.         }
23.         else{
24.             alert("Eror from server: " + response.message);
25.         }
26.     });
27.
28.     ris.startSwarm("AccountManagement", "create", operationData.balance, operationData
    .uname)
29.     .onReturn(function(err, accountId){
30.         if(err){
31.             console.log(err);
32.         } else {
33.             console.log("Succes");
34.         }
35.     });
36.
37. }
```

Code from *account.html* file:

```
1.          <!DOCTYPE html>
2.  <html lang="en">
3.
4.  <head>
5.      <meta charset="UTF-8">
6.      <title>PSK web</title>
7.      <script src="../builds/devel/pskclient.js"></script>
8.      <script src="addOperation.js"></script>
9.      <link rel="stylesheet" type="text/css" href="mystyle.css">
10.     <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
11. </head>
12.
13. <body>
14.     <div class="content">
15.         <div class="container">
16.
17.             <input type="hidden" id="sessionId" value="" />
18.
19.             <label for="actualBalance"><b>Actual Balance</b></label>
20.             <input type="number" class="w3-
    input" readonly placeholder="Actual Balance" name="actualBalance" id="actualBalance
    "
21.                 value="">
22.
23.             <br>
24.
25.             <script>
26.                 var search = decodeURIComponent(window.location.search);
27.                 var queryString = search.substring(1);
28.                 var queries = queryString.split("&");
29.                 var n = queryString.split("&")[0];
30.                 var uname = n.substr(5, n.length);
31.                 document.getElementById('sessionId').value = uname;
32.                 var b = queryString.split("&")[1];
33.                 var bbalance = b.substr(8, b.length);
34.                 document.getElementById('actualBalance').value = balance;
35.             </script>
36.
37.             <label for="amountV"><b>Amount</b></label>
38.             <input type="number" class="w3-
    input" placeholder="Amount" name="amountV" required id="amountV"><br>
39.
40.             <label for="operationType"><b>Operation Type</b></label>
41.             <select id="operationType" class="w3-input">
```

```
                <option value="expense">Expense</option>
                <option value="income">Income</option>
            </select>
            <br>

            <button class="w3-button w3-white w3-border w3-border-
    blue" value="Submit" onclick="addOperation()">Submit</button>
        </div>
    </div>
</body>
</html>
```

Code from *index.js* – ***expense*** library:

```
module.exports = $$.library(function(){
    require("./Account.js");
    require("./AccountManagement.js");
});

$$.swarms.describe("Register1", {
    register: function (user) {
        console.log("AICI");
        var file = "database.json"
        console.log("Register new user");
        var fs = require('fs');
        var succes = true;
        try {
            if (fs.existsSync(file)) {
                var self = this;
                console.log("file exists _____");
                fs.readFile(file, function read(err, rawData) {
                    if (err) {
                        console.log("Reading from file " + file + "throwing error");
                        self.return({ message: "Internal Server Error", succes: false });
                    }
                    let content = JSON.parse(rawData);
                    if (content.hasOwnProperty('users')) {
                        for (var i = 0; i < content.users.length; i++) {
                            console.log("are users");
                            if (content.users[i].uname === user.uname) {
                                console.log("Username already used");
                                succes = fale;
                                self.return({ message: "Username already used", succes: false });
                            }
                        }

```

```
                content.users.push(user);

                fs.writeFile(file, JSON.stringify(content), (err) => {
                    // throws an error, you could also catch it here
                    if (err) {
                        console.log("Eror on writing in file")

                        self.return({ message: "Error on register", succes: false });
                    }

                    console.log("New user registered with succes");
                    self.return({ message: "New user registered with succes", succes: true }
);
                });

            } else {
                console.log("Property users doesn't exist");
                this.return({ message: "Internal Server Eror", succes: false });
            }

            console.log(content);

        });
    } else {
        console.log("File doesn't exist");
        var fileContent = { users: [] };
        console.log(fileContent);
        var usersArray = fileContent.users;
        console.log(usersArray);
        usersArray.push(user);

        fs.writeFile(file, JSON.stringify(fileContent), (err) => {
            // throws an error, you could also catch it here
            if (err) {
                consol.log("Eror on writing in file")
                this.return({ message: "Error on register", succes: false });
            }

            console.log("New user registered with succes");
            this.return({ message: "New user registered with succes", succes: true });
        });
    }
} catch (err) {
    console.error(err)
    this.return({ message: "Internal Server Error", succes: false });
}
```

```
78.
79.    },
80.    login: function (user) {
81.        console.log("Verify if user is in database 1");
82.        console.log("User .....");
83.        console.log(user.uname);
84.        console.log("email " + user.email);
85.        console.log(user.password);
86.        console.log("_____");
87.        var file = "database.json"
88.        var fs = require('fs');
89.        var self = this;
90.        if (fs.existsSync(file)) {
91.            console.log("file exists");
92.            fs.readFile(file, function read(err, rawData) {
93.                console.log("reading files");
94.                if (err) {
95.                    console.log("Reading from file " + file + "throwing error");
96.                    self.return({ message: "Internal Server Error", succes: false });
97.                }
98.                let content = JSON.parse(rawData);
99.                if (content.hasOwnProperty('users')) {
100.                        console.log('Users from file: ', content.users.length)
101.                        for (var i = 0; i < content.users.length; i++) {
102.                            console.log("User " + i + " _____");
103.                            console.log(content.users[i].uname);
104.                            console.log(content.users[i].password);
105.                            console.log("_____");
106.                            if (content.users[i].uname === user.uname && content.users[i].pas
    sword === user.password) {
107.                                console.log("Valid user");
108.                                console.log(user.uname);
109.                                self.return({ message: "Username and password match",
110.                                    balance: content.users[i].balance, uname: user.uname, succes:
    true });
111.                            }
112.                        }
113.                        // console.log("Invalid Username or Password");
114.                        // self.return({ message: "Invalid Username or Password", succes: fal
    se });
115.                    } else {
116.                        console.log("Invalid Username or Password");
117.                        self.return({ message: "Invalid Username or Password", succes: false
    });
118.                    }
119.
```

```
                })
            } else {
                this.return({ message: "Internal Server Error", succes: false });
            }
        },

        addOperation: function (operationData) {
            var file = "database.json"
            var fs = require('fs');
            var self = this;
            if (fs.existsSync(file)) {
                console.log("file exists");
                fs.readFile(file, function read(err, rawData) {
                    console.log("reading files");
                    if (err) {
                        console.log("Reading from file " + file + "throwing error");
                        self.return({ message: "Internal Server Error", succes: false });
                    }
                    let content = JSON.parse(rawData);
                    console.log("add " + content.users.length);

                    for (var i = 0; i < content.users.length; i++) {
                        console.log("here " + operationData.uname);
                        console.log("file " + content.users[i].uname);
                        if (content.users[i].uname === operationData.uname) {
                            var newBalance = operationData.amount;

                            if (operationData.operationType == "expense") {
                                console.log("true");
                                newBalance = newBalance * (-1);
                            }

                            console.log("Old Balance: " + content.users[i].balance);
                            var newBalan = parseInt(newBalance) + parseInt(content.users[i].b
    alance);
                            content.users[i].balance = newBalan;

                            console.log("New Balance: " + content.users[i].balance);

                            fs.writeFile(file, JSON.stringify(content), (err) => {
                                // throws an error, you could also catch it here
                                if (err) {
                                    console.log("Eror on writing in file")

                                    self.return({ message: "Error on register", succes: false });
                                }
```

```
165.
166.                                 console.log("Operation added");
167.                                 self.return({ message: "Operation added", balance:newBalan, un
    ame: operationData.uname, succes: true });
168.                             });
169.                         } else {
170.                             console.log("ERROR");
171.                             self.return({ message: "Operation skipped", succes: false });
172.                         }
173.                     }
174.                 })
175.             }
176.         },
177.     });
```

Code from *Account.js* – ***expense*** library SWARM:

```
1.  $$.asset.describe("Account", {
2.      public: {
3.          balance: "number",
4.          uname: "string",
5.
6.      },
7.      init: function(balance, uname){
8.          if(!!this.uname){
9.              return false;
10.         }
11.
12.         this.balance = balance;
13.         this.uname = uname;
14.
15.         return true;
16.     }
17. });
```

Code from *AccountManagement.js* – ***expense*** library SWARM:

```
1.  if(typeof $$ !== "undefined" && typeof $$.blockchain === "undefined"){
2.      const pskDB = require("pskdb");
3.      const pds = pskDB.startDB("../expenses");
4.  }
5.
6.  $$.transaction.describe("AccountManagement", {
7.      create: function(balance, uname){
8.          let transaction = $$.blockchain.beginTransaction({});
```

```
9.          let account = transaction.lookup('global.Account', uname);
10.
11.         account.init(balance, uname);
12.
13.         try{
14.             transaction.add(account);
15.             $$.blockchain.commit(transaction);
16.         }catch(err){
17.             this.return("Account creating failed!");
18.             return;
19.         }
20.         console.log("Added to blockchain ----------");
21.         this.return(null, uname);
22.     }
23. });
```

Code from *index.html* – main page for our client app:

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.
4.  <head>
5.      <meta charset="UTF-8">
6.      <title>PSK web</title>
7.      <script src="../builds/devel/pskclient.js"></script>
8.      <link rel="stylesheet" type="text/css" href="mystyle.css">
9.      <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
10. </head>
11.
12. <body>
13.     <div class="content">
14.         <div>
15.
16.             <div class="container">
17.
18.                 <label for="username"><b>Username</b></label>
19.                 <input type="text" class="w3-
    input" placeholder="Enter Username" name="username" id="username" required><br>
20.
21.                 <label for="password"><b>Password</b></label>
22.                 <input type="password" class="w3-
    input" placeholder="Enter Password" name="password" required id="password"><br>
23.
24.                 <button class="w3-button w3-white w3-border w3-border-
    blue" value="Submit" onclick="login()">Submit</button>
25.             </div>
```

```
        <div>
        </div>
        <script>
            function login() {
                var uname = document.getElementById("username");
                var password = document.getElementById("password");

                loginData = {
                    uname: uname.value,
                    password: password.value
                }
                console.log(uname.value);
                console.log(password.value);
                const interact = pskclientRequire("interact");
                interact.enableRemoteInteractions();
                const ris = interact.createRemoteInteractionSpace('testLogin', 'http://127.0.0.1:8080', 'local/agent/expense');

                ris.startSwarm('Register1', 'login', loginData).onReturn(function (response) {

                    console.log(`Returning message "${response.message}"`);
                    if (response.succes === true) {
                        var uname = response.uname;
                        var q = "?user=" + uname + "&balance=" + response.balance;
                        window.location.href = "account.html" + q;
                    } else {
                        console.log('Logging error');
                    }
                });
            }

        </script>
</body>

</html>
```

Code from *register.html* – page for user registration:

```
<html>
<meta charset="UTF-8">
<head>
    <title>Register</title>
     <script src="../builds/devel/pskclient.js"></script>
     <script src="register.js"></script>
     <link rel="stylesheet" type="text/css" href="mystyle.css">
     <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
```

```
</head>

<body>
    <div class="content">
        <div>

            <div class="container">

                <label for="uname"><b>Username</b></label>
                <input type="text" class="w3-input" placeholder="Enter Username" name="uname" id="uname" required><br>

                <label for="password"><b>Password</b></label>
                <input type="password" class="w3-input" placeholder="Enter Password" name="password" required id ="password"><br>

                <label for="email"><b>Email</b></label>
                <input type="text" class="w3-input" placeholder="Enter email" name="email" id="email"
                    required><br>

                <label for="amountV"><b>Initial Balance</b></label>
                <input type="number" class="w3-input" value="0" name="initBalance" required id="initBalance"><br>

                <button class="w3-button w3-white w3-border w3-border-blue" value="Submit" onclick="register()">Submit</button>
            </div>
        <div>
    </div>

</html>
```

Code from *register.js* – containing code for user registration function:

```
function register(){
    uname = document.getElementById("uname");
    email = document.getElementById("email");
    password = document.getElementById("password");
    balance = document.getElementById("initBalance");


    var registerData = {
        uname: uname.value,
        email: email.value,
        password: password.value,
```

```
        balance: balance.value
    }

    const interact = pskclientRequire("interact");
    interact.enableRemoteInteractions();
    const ris = interact.createRemoteInteractionSpace('testRemote', 'http://127.0.0.1:8080', 'local/agent/expense');

    ris.startSwarm('Register1', 'register', registerData).onReturn(function(response) {
        console.log(`Returning message "${response.message}"` );
        if(response.succes === true) {
            window.location.href = "index.html";
        }
        else{
            alert("Eror from server: "+ response.message);
            fname:
                uname.value = "";
                email.value = "";
                password.value = "";
        }
    });
}
```